NESNE TABANLI PROGRAMLAMA - II
Nesne ve Sınıf Yapısı

Öğr. Gör. Dr. Murat ASLANYÜREK

Kırklareli Üniversitesi

Pınarhisar MYO

Nesneye Yönelik Programlamaya Giriş

- Nesneye yönelik programlama,
  - 1960'lı yıllarda ortaya çıkmıştır.
  - Yazılımdaki karmaşıklığı önler.
  - Programlar daha rahat kontrol edilir.
  - Oluşan hatalar, program parçalara bölündüğü için daha rahat tespit edilir.
  - Performansı arttır.
  - Büyük çaplı projelerde avantajları onu vazgeçilmez kılar.

- Yazılımdaki karışıklığa çözüm için kullanılnan bazı prensipler,
  - Kalıtım(Inheritance)
  - Çok Biçimlilik(Polymorphism)
  - Soyutlama(Abstraction)
  - Sarmalama(Encapsulution) -Kapsülleme

- Nesneye yönelik programlama mantığında hiyerarşi(sınıflar ve nesneler) vardır.

JAVA'DA NESNE VE SINIF YAPISI

- Java nesneye yönelik bir programlama dilidir. Nesneye yönelik bir programlama olmasından dolayı OOP'nin temel konseptlerini destekler.
- Bu konseptler : Çok biçimlilik (Polymorphism), Kalıtım (Inheritance), Kapsülleme (Encapsulation), Soyutlama (Abstraction), Sınıflar (Classes), Nesneler (Objects), Instance (Örnek), Metod, Mesaj Ayrıştırma olarak sıralanabilir.

- Doğadaki her şey bir nesnedir. Uçan kuş, evimizdeki kapı, pencere, dışardaki arabalar vs.
- Nesnelerin 2 özelliği vardır. **Durum** ve **Davranış**.

- Araba = Nesne,
  - Arabanın rengi, fiyatı, markası vb. **durumunu** belirtir.
  - Arabanın hızlanması, yavaşlaması, vites değiştirmesi vb. onun **davranışını** belirtir.
    - Ayrıca davranışa, attribute(özellik) denilebilir.

- Nesne: Nesneler durum ve davranışlara sahiptir. Örneğin bir kedinin durumları vardır : aç ya da tok olmaları, isimlerinin olması, yaşlarının olması, renklerinin olması vs. Ve kedilerin bir de davranışları vardır : miyavlamaları, uyumaları, korkmaları vs.
- Sınıf : Sınıf ise nesnelerin durum ve davranışlarının tiplerine göre tanımlandığı yapıdır diyebiliriz. Yani kedilerin bir renge sahip olma durumunu, yaşlarının belirleniyor olma durumunu, havlamak yerine miyavlıyor olmalarını vs. sınıf adı verdiğimiz şablon içerisinde tanımlarız.

JAVA'DA NESNELER

- Gerçek dünyadaki nesneleri ele alırsak hepsinin durum ve davranışlara sahip olduğunu söyleyebiliriz. Örneğin insanlar, arabalar, ağaçlar hep bir duruma ve davranışa sahip nesnelerdir. Bir insanı ele aldığımızı düşünelim. Bu insanın bir adı vardır, bu insan koşar, yüzer, uyur vs. Java'daki nesne yapısı da gerçek dünyadaki nesne yapısına oldukça benzer. Java nesnelerinin de durum ve davranışları vardır. Bir Java nesnesinde durumlar alanlarda tutulur ve davranışlar da metodlarda tasvir edilir.

NEW Anahtar Kelimesi ile Nesne Oluşturma

- Bir sınıftan oluşturulan her örneğe **nesne** denir. Örneğin, Insan sınıfından 10 adet insan oluşturduk, oluşturulan her insan nesnesinin boyu, yaşı vs. farklı olabilir.
- Javada bir sınıftan nesne oluşturmak için **new** anahtar söcüğü kullanılır.
  - Insan Mehmet = **new** Insan();

```java
package Sinif_Nesne_Kavrami;

public class O1_Insan {
    //özelliikler tanımlandı.( boy,cinsiyet, yas)
    double boy;
    String cinsiyet;
    int yas;

    public static void main(String[] args)
    {
        // insan sınıfından kisi adında bir nesne oluşturuldu
        O1_Insan kisi = new O1_Insan();

        // nesnenin boy değişkenine değer atandı
        kisi.boy= 1.78;
        System.out.println(kisi.boy);
    }
}
```

Console | Problems | Debug Shell | Search

```
<terminated> O1_Insan [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe
1.78
```

*Oluşturulan bir nesnenin özelliklerine **nokta(.)** ile erişiriz.*

***Nokta(.)** operatörü, bir sınıfın altında bulunan metodlara değişkenlere erişmemize olanak tanır.*

```java
package Sinif_Nesne_Kavrami;

public class O2_Araba {
    String marka, renk; //durum-özellik(attribute)
    void gazaBas() //davranış(metod)
    {
        System.out.println("Araba Hızlanıyor");
    }

    void freneBas() //davranış(metod)
    {
        System.out.print("Araba Yavaşlıyor");
    }
    public static void main(String[] args)
    {
        // Araba sınıfınfan a1 nesnesi oluşturuldu
        O2_Araba a1 = new O2_Araba();

        a1.marka="xxx";
        a1.renk="siyah";
        a1.gazaBas();
        a1.freneBas();
    }
}
```

Console | Problems | Debug Shell | Search

```
<terminated> O2_Araba [Java Application] C:\Program Files\Java\jdk-13.0.1\bir
Araba Hızlanıyor
Araba Yavaşlıyor
```

- **Not:** *Oluşturulan her nesne, bellekte farklı bir alanda tutulur. Nesnelerin bellekte tutulduğu bölgeye **heap** alanıdır. Bu nesnelere, o nesnenin **referansı** ile erişebiliriz. Bir nesnenin birden fazla referansı olabilir.*

JAVA'DA SINIFLAR

- Sınıflar, yaratılacak olan nesnelerin planlarıdır. Aşağıda bir sınıf örneği verilmiştir :

```
1  package Sinif_Nesne_Kavrami;
2
3  public class O3_Kedi {
4      String turu="XXX";
5      int yasi=2;
6      String rengi="Beyaz";
7
8      void miyavlama(){
9          System.out.println("Kedi miyavladı");
10
11     }
12
13    void hareketEtme(){
14        System.out.println("kedi hareket etti");
15
16    }
17    public static void main(String[] args){
18
19        O3_Kedi kedi = new O3_Kedi();
20        System.out.println("Türü: "+kedi.turu+" olan "
21            + kedi.yasi+ " yasindaki "
22                + kedi.rengi);
23        kedi.miyavlama();
24    }
25 }
```

```
Console  Problems  Debug Shell  Search
<terminated> O3_Kedi [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe
Türü: XXX olan 2 yasindaki Beyaz
Kedi miyavladı
```

```java
package Sinif_Nesne_Kavrami;

public class O4_Metotlar {

    int carp(int sayi1, int sayi2)
    {
        return sayi1*sayi2;
    }
    int topla(int sayi1, int sayi2)
    {
        return sayi1+sayi2;
    }

    public static void main(String[] args)
    {
        O4_Metotlar  islem1 = new O4_Metotlar ();
        O4_Metotlar  islem2 = new O4_Metotlar ();

        int sonuc1 = islem1.carp(6,12);
        int sonuc2 = islem2.topla(7, 1);
        System.out.println("Çarpma Sonucu= "+sonuc1);
        System.out.println("Toplama Sonucu= "+sonuc2);
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O4_Metotlar [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw

```
Çarpma Sonucu= 72
Toplama Sonucu= 8
```

Not: *Bir sınıf içerisinde istediğimiz kadar **metod** tanımlayabiliriz ve bunlara İstediğimiz kadar **parametre** gönderebiliriz.*

Yerel, Sınıf ve Nesne Değişkenleri

- Bir sınıf aşağıdaki değişken tiplerini barındırabilir:
  - Yerel Değişkenler: Metodların, constructor (yapıcı)'ların veya blokların içinde tanımlanmış değişkenlere yerel değişkenler denir. Değişken metodla birlikte başlatılır ve metod tamamlandığında yok edilir.
  - Nesne Değişkenleri: Bir sınıfın içinde tanımlı olan ama bir metodun, constructor'ın ya da bloğun içinde tanımlanmamış değişkenlerdir. Bu değişkenler sınıf başlatıldığında başlatılır. Bu değişkenlere tanımlı olduğu sınıfın metodları, constructor'ı ya da tanımlanmış olan blokları erişebilir.
  - Sınıf Değişkenleri: Bu değişkenler metod, constructor ve blok dışında tanımlanmış ve **static** anahtar kelimesiyle tanımlanmış değişkenlerdir.

Sınıf ve Nesne Degiskeni Örnek

Sınıf Değişkeni

```java
package Sinif_Nesne_Kavrami;


public class O5_SınıfDegiskeni {

    static int deger=5; // sınıf değişkeni

    public static void main(String[] args)
    {
        O5_SınıfDegiskeni d1 = new O5_SınıfDegiskeni();
        O5_SınıfDegiskeni d2 = new O5_SınıfDegiskeni();

        System.out.println(d1.deger);
        System.out.println(d2.deger);
    }
}
```

Console | Problems | Debug Shell | Search

```
<terminated> O5_SınıfDegiskeni [Java Application] C:\Program Files\Java\jdk-13.0
5
5
```

Nesne Değişkeni

```java
package Sinif_Nesne_Kavrami;

public class O6_NesneDegiskeni {

    int deger; // nesne değişkeni
    public static void main(String[] args)
    {
        O6_NesneDegiskeni d1 = new O6_NesneDegiskeni();
        O6_NesneDegiskeni d2 = new O6_NesneDegiskeni();

        d1.deger=7;
        d2.deger=9;

        System.out.println(d1.deger);
        System.out.println(d2.deger);
    }
}
```

Console | Problems | Debug Shell | Search

```
<terminated> O6_NesneDegiskeni [Java Application] C:\Program Files\Java\jdk-13.0
7
9
```

Sınıf ve Nesne Değişkenleri

- **Not:** *Nesne değişkenleri **dinamik değişkenler**, yani **instance variable** olarak da adlandırılır. Sınıf değişkenleri ise **static değişkenler** olarak adlandırılırlar.*

- **Not:** *Eğer nesne değişkeni tanımlarsak oluşturulan **her nesne için** bellekte bir yer ayrılır. Bu nesne değişkenlerinin değeri, programın herhangi bir yerinde değiştirilebilir. Sınıf değişkenleri için bellekte sadece **bir yer** ayrılır.*

- **Not:** *Sınıf değişkenleri **static** olarak tanımlandığı için bu değişkenlere nesne oluşturulmadan da sadece sınıf adını kullanarak ta erişilebilir. Yani bunlar **nesneden bağımsızdır**.*

- Sınıf değişkenleri aynı zamanda **static** değişkenler olarak adlandırılırlar.
- Sınıf değişkenleri, **global değişken** kavramına benzer.
- ***Static*** *olan değişkenlere sadece* ***sınıf adı*** *ile erişebiliriz.*
- *Sınıf değişkenleri* ***program bittiğinde*** *bellekten silinirler. Ve her program için yalnızda bir defa oluşturulurlar. Nesne değişkenleri ise* ***nesne yok olduğunda*** *bellekten silinirler.*

Static ile erişim

```java
package Sinif_Nesne_Kavrami;

public class O6_Static {

     String birim="PMYO";
     static String bolum ="Bilgisayar Programcılığı";

    public static void main(String[] args)
    {

        //System.out.print(birim); hatalı nesne olusturulmalı

        O6_Static n = new O6_Static(); //nesne oluştu

        System.out.println(n.birim); // nesne ile erişildi

        // nesne oluşturmandan erişildi
        System.out.print(bolum);
//Farklı bir sınıfta olsaydı sınıf ismiyle erişilebilirdi
    }

}
```

Console | Problems | Debug Shell | Search

```
<terminated> O6_Static [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe
PMYO
Bilgisayar Programcılığı
```

Pass By Value – Pass By Reference (Değer ve Referans Tipleri)

- Bir metoda parametre gönderdiğimiz varsayalım. Metod, bu parametreyi 10 ile çarpıyor ve ekrana bir şeyler yazdırıyor. Peki metoddan çıktığımızda asıl değişkenin değeri değişir mi?
  - Primitive(int, double, vb) tiplerde değer metoda parametre olarak gönderildiğinde değerin kopyası gönderilir. Orijinal değer değişmez.
  - Referans tiplerde, metoda değerin kendisi gönderilir. Metodda olan değişiklik değişkenede yasnır.

Primitive Tipler - Pass By Value

Primitive Tipler

```java
package Sinif_Nesne_Kavrami;

public class O7_PrimitiveTipler {
    public static void main(String[] args)
    {
        int a=5;
        System.out.println("Eski Değer = "+a);
        degistir(a);
        System.out.println("Yeni Değer = "+a);
    }
    static void degistir(int a)
    {
        a=a+10;
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O7_PrimitiveTipler [Java Application] C:\Program Files\J

```
Eski Değer = 5
Yeni Değer = 5
```

- Bu örnekte a değişkenini ilk olarak ekrana yazdırdık. Metotta bu değeri 10 ile topladık. Metototan çıktığımızda yeni a değerini ekrana yazdırdık. Asıl değerde bir değişiklik olmadı.
- Çünkü primitive tiplerde bir metoda değişkeninin değerinin kopyası gönderilir.

Referans Tipler - Pass By Reference

Referans Tipler

```java
package Sinif_Nesne_Kavrami;

public class O8_ReferansTipler {
    int x;
    public static void main(String[] args)
    {
        O8_ReferansTipler t = new O8_ReferansTipler();
        t.x=5;
        System.out.println("Eski Değer = "+t.x);
        t.degistir(t);
        System.out.println("Yeni Değer = "+t.x);
    }
     void degistir(O8_ReferansTipler t)
    {
        t.x=t.x+10;
    }
}
```

Console | Problems | Debug Shell | Search

```
<terminated> O8_ReferansTipler [Java Application] C:\Program Files\Java\jdk-13.0
Eski Değer = 5
Yeni Değer = 15
```

- Referans tipler new anahtar sözcüğü ile oluşturulur. Metoda parametre olarak nesnenin adresi gönderilir. Bellekte aynı yeri işaret ettikleri için orijinal değeri değişir.
- Not: dizilerde referans tipler olarak değerlendirilir.

- Dizilerimizde bir referans tipi olduğundan bir örnek üzerinden inceleyelim;

```java
package Sinif_Nesne_Kavrami;

public class O9_ReferansTiplerDizi {
    public static void main(String[] args) {
        int array[]= {5,6,7,1};
        System.out.println("Dizinin eski 0. idisi= "+array[0]);
        fonksiyon(array);
        System.out.println("Dizinin yeni 0. idisi= "+array[0]);
    }
    static void fonksiyon(int array[]) {
        array[0]*=6;
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O9_ReferansTiplerDizi [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\j

```
Dizinin eski 0. idisi= 5
Dizinin yeni 0. idisi= 30
```

KAYNAKLAR

- KİRAZLI, M., ‘‘ Java 7:Yeni Başlayanlar için’’, Kodlab, 8.Baskı, Ekim,2015.
- https://javaplanet.wordpress.com/2017/03/27/javada-sinifclass-yapisi/
- http://teknokafe.net/2017/09/19/java-oop-deger-ve-referans-tipleri-yapicilar-constructor/